A STUDY ON APPROXIMATION ALGORITHMS FOR MATCHING IN GRAPHS AND HYPERGRAPHS

Synopsis

Submitted in the partial fulfillment for the requirement of

Doctor of Philosophy

In

Mathematics

Submitted by

Urvashi

Under the supervision of

Prof. Gur Saran
Department of Mathematics

Department of Mathematics, Faculty of Science
Dayalbagh Educational Institute (Deemed University)
Dayalbagh, Agra-282005

2022

1. Introduction

Several problems of different domains can be modeled as problems on graphs. In computer science graphs, for instance, are used in the representation and analysis of communication network and data organization [RA]. Many problems related to graphs lie in the category of the optimization problems [BAS], especially discrete optimization problems. Most of the discrete optimization problems are NP-hard for which there are no efficient algorithms that can find (1) an exactly optimal solution (2) to all instances of the problem (3) in polynomial time in the size of the input, unless $P = NP$ [Das]. Therefore, if $P \neq NP$ then for dealing with NP-hard optimization problems, at least one of the above mentioned requirements must be relaxed to find an efficient algorithm. The approach used by approximation algorithms is a way to relax the requirement of finding an optimal solution in polynomial time [WS].

2. Approximation algorithms

It was David S. Johnson (1974) [Joh], who introduced the term "approximation algorithm". This type of algorithm is used when one is trying to find a solution that closely approximates the optimal solution in terms of its value. In other words, an algorithm is said to be an approximation algorithm for given optimization problem if for any given instance, it returns an approximate solution which is a feasible solution [Das]. A formal definition is as follows [WS]:

> "An α-approximation algorithm for an optimization problem is a polynomial time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution, where the approximation factor may depend on the input size".

Here α is the performance guarantee of the algorithm. It is also called approximation ratio or approximation factor of the algorithm with the convention that $\alpha > 1$ for minimization problems and $\alpha < 1$ for maximization problems.

Approximation algorithms have been used to solve many optimization problems [GM] like problems related to matching in graphs and hypergraphs [EKMS].

3. Matching in graphs

The problem of finding a matching is one of the fundamental problems in graph theory. Consider a simple graph (without loops and parallel edges) $G = (V, E)$ where V is the vertex set and E is the set of edges of G. A matching M in G is the set of non-loop edges with no shared endpoints or vertices [Wes]. The vertices incident to the edges of a matching M are said to be saturated by M; otherwise vertices called unsaturated or a vertex is said to be a free vertex if it does not appear in any edge in M. A matching is said to be perfect matching if it saturates every vertex in the graph. A graph is said to be bipartite if its vertex set can be expressed as a union of two disjoint independent sets. Similarly, a graph is said to be tripartite if its vertex set can be decomposed into three disjoint independent sets. In a bipartite graph, perfect matching expresses a bijection from one partition set to another [Wes].

The property of matching in graphs is inherent in numerous real-world applications [LR], such as in the pairing problem where for a group of people, some people are well-suited to be roommates and some pairs not. Here the question arises, in what conditions one can pair them all up. The solution of this pairing problem can be obtained by finding a matching [Wes]. Another application is hospital residency programs for which interns needs to be matched. Matching is also used to save lives by finding a compatible match in kidney transplants [SGW$^+$].

3.1 Study of Berge's theorem

A matching M is an inclusion-maximal matching if $M \cup \{e\}$ is no longer a matching for $e \in E \setminus M$ [EKMS]. A maximum matching is matching of maximum size among all matchings in a graph where the size of a matching is the number of edges in it. Every maximum matching is a maximal matching, but the converse need not hold. For a given matching M, an M-alternating path is a path that alternates between edges in M and edges not in M and if this path has endpoints as a free vertex then this path is known as an M-augmenting path. Given an M-augmenting path P, one can replace the edges of M in P with the remaining edges of P to obtain a new matching M' with one more edge [Wes]. This fact leads to the famous Berge's theorem [Ber57] which is stated as:

> "A matching M is a maximum matching in a graph G if and only if there is no M-augmenting path in G".

This result is used by most of the matching algorithms for finding a maximum matching where M-alternating paths are explored to identify M-augmenting paths. An inclusion-maximal matching is a 2-approximation of maximum matching [EKMS]. Eggert et al. [EKMS] used the property of this theorem for finding a maximum matching in a bipartite graph in the semi-streaming model. For this problem, they used augmenting paths for increasing the size of matching and achieved deterministic $(1+\epsilon)$ approximation algorithm which requires only $O((1/\epsilon)^5)$ passes over the input stream.

4. Hypergraphs

A hypergraph is a generalization of the concept of a graph [Bec]. A hypergraph $H = (V, E)$ is a finite set of vertices $V = [n] = \{v_1, \dots, v_n\}$ with a finite family of hyperedges $E = [m] = \{e_1, \dots, e_m\}$ where each $e \in E$ is called hyperedge which is a non-empty subset of V. The size or cardinality of hyperedge e in E is the number of vertices in e. It is possible that a vertex does not contained in any hyperedge, such a vertex is called isolated. But in some cases it is seen that isolated vertices are not allowed in the definition of hypergraph [Ber84]. Two vertices of a hypergraph are said to be adjacent if there exists an edge containing these vertices. A vertex *v* and an edge *e* of a hypergraph are said to be incident if *v* is a member of E. If $|e| = k$ for each $e \in E$ then the hypergraph is said to be a k-uniform hypergraph. If no subset $V' \subset V$ spans more than $p|V'| - q$ hyperedges then the hypergraph $H = (V, E)$ is called (p, q)-sparse hypergraph. Here the span of V' defines the set of edges whose end points in V' and p and q are sparsity parameters [ST].

In some kind of applications, hypergraphs are more suitable than graphs. For example, in railway vehicles rotation planning [BHK+] and in biological network modeling and its analysis, the hypergraph plays a significant role [KHT]. Many optimization problems which are solvable in polynomial time on graphs are NP-hard on hypergraphs, i.e., finding b-matching with maximum weight [KS].

4.1 Matching in hypergraphs

In a hypergraph, a matching M is a set of hyperedges such that every vertex is contained in at most one of the hyperedges of M. If hyperedges of M contain all the vertices of hypergraph then it is called perfect matching. A vertex which is contained in an hyperedge of M is said to be covered by M,

otherwise it is said to be exposed by M [Bec]. Let $w : E \to [0,1] \cap \mathbb{Q}$ be a weight function assigned on hyperedges of E and $w(e)$ be the weight of a hyperedge $e \in E$. The weight of each set $X \subseteq E$ is denoted by $w(X)$ such that $w(X) = \sum_{e \in X} w(e)$. We can generalize the above definition of matching for hypergraphs. Call a set $M \subseteq E$ a b-matching if no vertex is contained in more than b edges from M where $b \in \mathbb{N}_{\geq 1}$ [KS]. A problem of finding b-matching with maximum weight is known as maximum b-matching [KS]. Here maximum weight is denoted by OPT_b. Let the maximum hyperedge cardinality be donated by l such that $l = \max_{e \in E} |e|$. This problem can be written as in the packing integer program (PIP) [KS]:

$$\max\left\{\sum_{j=1}^{m} w_j x_j \, ; Ax \leq b, x \in \{0,1\}^m\right\}$$

Here, A is $n \times m$ vertex-edge incidence matrix of the hypergraph with entries 0 or 1, i.e., $A_{ij} = 1$ if and only if i^{th} vertex is contained in the j^{th} edge; and $x_j = 1$ if and only if j^{th} edge is selected for the b-matching. In combinatorics and optimization, maximum b-matching is a classical problem for $b = 1$ and for general value of b. It is also used in generalized setting of combinatorial auctions [GL, Kry].

4.2 Literature review for b-matching

Theory of matching involves many combinatorial problems in hypergraphs. Several combinatorial problems can be reconstructed in the form of matching in hypergraph case. Therefore, matching theory receives most attention in the case of hypergraphs than graphs. The problem of finding maximum weight matching in context of graphs is polynomial time solvable [Edm], [Bec] whereas problem of finding b-matching with maximum weight is NP-hard in hypergraphs even if maximum edge cardinality l restricted to at most 3 [KS]. Therefore, approximation algorithms are used to solve it. For maximum b-matching, approximation algorithm is called α approximation if it gives solution with weight at least α OPT_b for $0 < \alpha \leq 1$, as b-matching with maximum weight is a maximization problem. Thus, approximation guarantee or performance guarantee becomes better as α comes close to 1.

For $b = 1$, which means for hypergraph 1-matching, Gonen and Lehmann [GL] exhibited that no polynomial time approximation factor is better than $n^{1/2-\epsilon}$ for $\epsilon > 0$, unless $NP = ZPP$. (ZPP or

zero-error probabilistic polynomial time refers the class of problems which are recognized by probabilistic Turing machine, that always return correct answer and solvable in polynomial running time). They originally proved this result for Weighted Multi-Set Packing (WMSP) problem. Another result was proved by Hazan et al. [HSS] stated as there no polynomial time $\Omega(k/\ln k)$ approximation algorithm exists, unless $P = NP$ for k-uniform hypergraphs. Krysta [Kry] used oblivious greedy algorithm and accomplished an approximation of $(\sqrt{n}+1)^{-1}$ and $(l+1)^{-1}$ independently of b.

For large b, specifically $b = \Omega(\epsilon^{-2}, \ln n)$ for $\epsilon > 0$, Srivastav et al. [SS] showed that randomized rounding attains constant factor approximation of $1-\epsilon$ with at least 0.73 probability. Another improved result based on randomized rounding on fractional solution proposed by Srinivasan [Sri] uses a derandomized algorithm which gives approximation ratio $O(\min\{n^{1/b+1}, l^{1/b}\})$ for any b by solving an LP relaxation. But for small b namely $b \ll \ln n$, this approximation does not work efficiently. This ratio becomes constant for $b = \Omega(\ln n)$. After that, Kliemann et al. [KS] proposed a hybrid algorithm which combines randomized rounding with a greedy algorithm. Experiments showed that this hybrid algorithm for $b > 1$, attains an approximation of 0.75 to 1.0 which is better than oblivious or randomized rounding algorithm. El Ouali et al. [EFS] showed by generalizing the result of Hazan et al.[HSS] that for k-uniform hypergraphs, for every $2 \leq b \leq k/\log k$ there is no polynomial-time approximation within any ratio smaller than $\Omega(k/b \log k)$, unless $P = NP$. But even now the range $k/\log k \leq b \leq \log n$ needs to be explored.

5. Algorithm engineering

Algorithm engineering is an area which deals with the gap between the algorithms that are developed for algorithmic theory and the algorithms that deal with the real-world instances because it is seen that asymptotically faster algorithms need not be well suited for practical purposes. It is also used to convert algorithmic information to practical application in fast manner [CK].

The algorithm engineering (AE) cycle, see Fig.1, given by Peter Sanders [San] consists mainly of algorithmic design, analysis, implementation and experimental evaluation. This main cycle is driven by falsifiable hypotheses which are validated by experiments, although there are some other aspects of AE outside the main cycle.

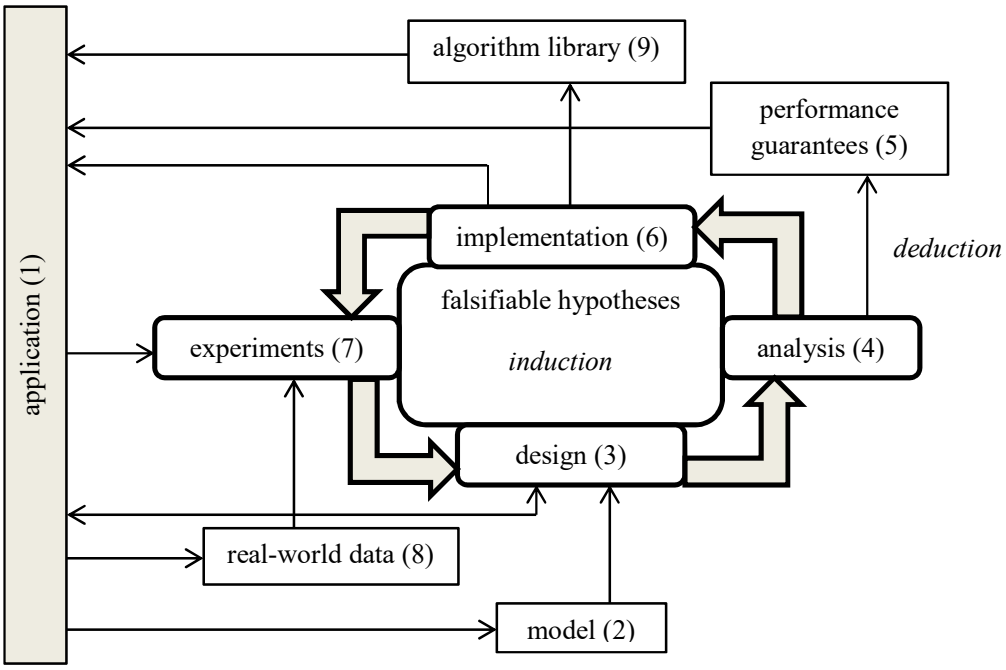

Fig. 1. Algorithm engineering cycle

AE cycle focuses on algorithmic development. It starts with some particular application (1) in mind and after that search for its realistic model (2) which gives relevant and possible solution. The main cycle starts with initial algorithmic design (3) for solving a considered model. Further step is to theoretical analysis (4) of this design of algorithm and obtain performance guarantees (5) such as approximation factor or ratios, asymptotic runtime etc. After this, the next step is to implement (6) the considered algorithm. This step may be time consuming but on the other hand it is the most significant step of AE. Success of this step depends on the reasonability of algorithm in its implementation complexity because the transformation process of purely theoretical algorithmic concepts into actual code not always clear. For many practical problems, the input for algorithms should be stored in flexible format such as linked list instead of array. In case of graphs, if input is in the form of compressed adjacency matrix then sometimes for full adjacency matrix, edge look-ups is not easy to perform in constant time. There may be the case that data is too large to store. Then to resolve this issue an alternative algorithm which provide constant time edge look-ups but are asymptotically slower, may become beneficial. Such type of observations can be done by experiments (7). For considered application, real-world data (8) should be used whose input instances must be at

least similar to problems for such application. Using this data, one can compare different algorithms that are beneficial for practice. By looking results of experiments, further improvement can be made in algorithmic design. Finally, the output of AE cycle should be some type of algorithm library (9) which consists of reusable code and its results can be further used to bridge the gap between algorithmic theory and practical aspects [CK].

AE has many interesting applications in various domains such as in information retrieval, communication networks and geographic information systems and in engineering where [San] described methodology of AE on external minimum spanning tree (MST) [DSSS] which is used for image segmentation.

6. Unique games conjecture

The unique games conjecture (UGC) was proposed by S. Khot [Kho02] in 2002. It is a hardness assumption which states that it is hard to approximate a certain combinatorial optimization problem in a strong sense. The key objective for the conjecture is to prove inapproximability results for *NP*-complete problems.

Unique games conjecture [Kho10]: For any ε, $\delta > 0$, there exists a constant $n = n(\varepsilon, \delta)$, such that given a unique game instance $\mathcal{U}(G(V,E), [n], \{\pi_e | e \in E\})$, it is NP-hard to distinguish between the following two cases:

i. YES Case: $\mathrm{OPT}(\mathcal{U}) \geq 1 - \varepsilon$.

ii. NO Case: $\mathrm{OPT}(\mathcal{U}) \leq \delta$.

here unique game $\mathcal{U}(G(V,E), [n], \{\pi_e | e \in E\})$ is a constraint satisfaction problem where $G(V,E)$ is a directed graph whose vertices represent as variables and edges represent as constraints. The objective is to assign to each vertex a label from the set $[n]$. The constraint on an edge $e = (u,v) \in E$ is described by a bijection $\pi_e : [n] \mapsto [n]$. A labeling $L : V \mapsto [n]$ satisfies the constraint on edge $e = (u,v)$ if and only if $\pi_e\big(L(u)\big) = L(v)$. Assuming $\mathrm{OPT}(\mathcal{U})$ denotes the maximum fraction of constraints that can be satisfied by any labeling such that

$$\mathrm{OPT}(\mathcal{U}) = \max_{L:V \mapsto [n]} \frac{1}{|E|} |\{e \in E \mid L \text{ satisfies } e\}|$$

It is seen that unique game is a problem to find an optimal labeling to a unique game instance.

7. Objectives

1. To study approximate Berge's theorems and its consequence for matching algorithms.
2. To explore the possibility of application of Berge's theorem for tripartite graphs.
3. To study approximation algorithms for b-matching in hypergraphs for any $b << \log(n)$ where n is the number of vertices. Is there an approximation with guarantee at least $\alpha\ \mathrm{OPT}_b$, $\alpha \in (0,1)$ constant possible?
4. To extend objective 3 to sparse hypergraph, i.e., bounded co-degree etc.
5. To do experimental studies and algorithm engineering for third and fourth objectives.
6. To study Non-Approximability for objectives 3 and 4, if $P \neq NP$ or other complexity measures conjectured to be true including the unique games conjecture.

References

[BAS] Baghel, M., Agrawal, S., Silakari, S.: Survey of metaheuristic algorithms for combinatorial optimization. International Journal of Computer Applications 58(19), 21-31 (2012)

[Bec] Beckenbach, I. L.: Matchings and Flows in Hypergraphs. Dissertation (2019)

[Ber57] Berge, C.: Two theorems in graph theory. Proceedings of the National Academy of Sciences 43(9), 842-844 (1957)

[Ber84] Berge, C.: Hypergraphs: Combinatorics of Finite Sets. vol. 45. Elsevier (1984)

[BHK+] Borndörfer, R., Heismann, O., Karbstein, M., Liebchen, C., Möhring, R., Möller, A., Römisch, W.: Traveling efficiently with mathematics. Deuflhard, P.; Grötschel, M.; Hömberg, D.; 3Es könnte sogar sein, dass die Bestandslösung für eine geringfügig andere Gewichtung der Zielfunktion optimal ist (2013)

[CK] Chimani, M., Klein, K.: Algorithm engineering: Concepts and practice. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss. M. (eds.) Experimental methods for the analysis of optimization algorithms. pp. 131-158. Springer, Heidelberg. (2010)

[Das] Das, G. K.: Introduction to Approximation Algorithms. Available: http://www.iitg.ac.in/gkd/aie/slide/approximation-algorithms.pdf (2016)

[DSSS] Dementiev, R., Sanders, P., Schultes, D., Sibeyn, J.: Engineering an external memory minimum spanning tree algorithm. In: Exploring New Frontiers of Theoretical Informatics. pp. 195-208. Springer, MA (2004)

[Edm] Edmonds, J.: Path, Trees and Flowers. Canadian Journal of Mathematics 17(3), 449-467 (1965)

[EFS] El Ouali, M., Fretwurst, A., Srivastav, A.: Inapproximability of b-Matching in k-Uniform Hypergraphs. In: Katoh, N., Kumar, A. (eds.) WALCOM 2011. LNCS, vol. 6552, pp. 57-69. Springer, Heidelberg (2011)

[EKMS] Eggert, S., Kliemann, L., Munstermann, P., Srivastav, A.: Bipartite matching in the semi-streaming model. Algorithmica 63(1-2), 490-508 (2012)

[GL] Gonen, R., Lehmann, D. J.: Optimal solutions for multi-unit combinatorial auctions. Branch and bound heuristics. CoRR cs.GT/0202032, pp. 13-20 (2002); Presented at ACM Conference on Electronic Commerce (2000)

[GM] Gärtner, B., Matoušek, J.: Approximation Algorithms and Semidefinite Programming. Springer Science and Business Media (2012)

[HSS] Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k-set packing. Computational Complexity 15(1), 20-39 (2006)

[Joh] Johnson, D. S.: Approximation Algorithms for Combinatorial Problems. Journal of Computer and System Sciences 9, 256-278 (1974)

[Kho02] Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, 767-775 May 2002

[Kho10] Khot, S.: On the Unique Games Conjecture (Invited Survey). Paper presented at 25th Annual IEEE Conference on Computational Complexity, 99-121 (2010)

[KHT] Klamt, S., Haus, U. U., Theis, F.: Hypergraphs and cellular networks. PLoS computational biology 5(5) (2009)

[Kry] Krysta, P.: Greedy approximation via duality for packing, combinatorial auctions and routing. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 615–627. Springer, Heidelberg (2005)

[KS] Kliemann, L., Srivastav, A.: Experimental Study of Non-oblivious Greedy and Randomized Rounding Algorithms for Hypergraph b-Matching. In: Vahrenhold, J. (eds.) SEA 2009. LNCS, vol. 5526, pp. 185-196 Springer, Heidelberg (2009)

[LR] Leighton T., Rubinfeld R.: Graph theory II. Available: https://www.cs.princeton.edu/courses/archive/fall06/cos341/handouts/graph2.pdf (2006)

[RA] Riaz, F., Ali, K. M.: Applications of graph theory in computer science. Paper presented at 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, IEEE 142-145 July 2011

[San] Sanders, P.: Algorithm engineering–an attempt at a definition. In: Efficient algorithms, pp. 321-340. Springer, Heidelberg (2009)

[SGW+] Segev, D. L., Gentry, S. E., Warren, D. S., Reeb, B., Montgomery, R. A.: Kidney paired donation and optimizing the use of live donor organs. Jama 293(15), 1883-1890 (2005)

[Sri] Srinivasan, A.: Improved approximation guarantees for packing and covering integer programs. SIAM Journal on Computing 29(2), 648-670 (1999)

[SS] Srivastav, A., Stangier, P.: Tight approximations for resource constrained scheduling and bin packing. Discrete Applied Mathematics 79, 223-245 (1997)

[ST] Streinu, I., Theran, L.: Sparse hypergraphs and pebble game algorithms. European Journal of Combinatorics 30(8), 1944-1964 (2009)

[Wes] West, D. B.: Introduction to Graph Theory. vol. 2. Upper Saddle River: Prentice hall (2001)

[WS] Williamson, D. P., Shmoys, D. B.: The Design of Approximation Algorithms. Cambridge university press (2011)